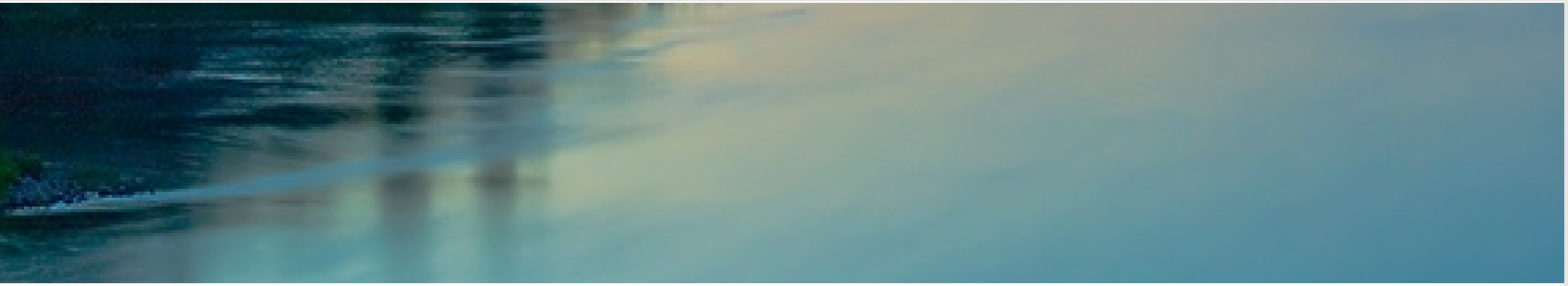


Neural Networks

Prof. Dr. Rudolf Kruse

Computational Intelligence Group
Faculty for Computer Science
kruse@iws.cs.uni-magdeburg.de

General (Artificial) Neural Networks

Basic graph theoretic notions

A (directed) **graph** is a pair $G = (V, E)$ consisting of a (finite) set V of **nodes** or **vertices** and a (finite) set $E \subseteq V \times V$ of **edges**.

We call an edge $e = (u, v) \in E$ **directed** from node u to node v.

Let $G = (V, E)$ be a (directed) graph and $u \in V$ a node. Then the nodes of the set

$$\mathrm{pred}(u) = \{v \in V \mid (v, u) \in E\}$$

are called the **predecessors** of the node u
and the nodes of the set

$$\mathrm{succ}(u) = \{v \in V \mid (u, v) \in E\}$$

are called the **successors** of the node u.

General definition of a neural network

An (artificial) **neural network** is a (directed) graph $G = (U, C)$, whose nodes $u \in U$ are called **neurons** or **units** and whose edges $c \in C$ are called **connections**.

The set U of nodes is partitioned into

- the set U_{in} of **input neurons**,
- the set U_{out} of **output neurons**, and
- the set U_{hidden} of **hidden neurons**.

It is

$$U = U_{\text{in}} \cup U_{\text{out}} \cup U_{\text{hidden}},$$

$$U_{\text{in}} \neq \emptyset, \qquad U_{\text{out}} \neq \emptyset, \qquad U_{\text{hidden}} \cap (U_{\text{in}} \cup U_{\text{out}}) = \emptyset.$$

General Neural Networks

Each connection $(v, u) \in C$ possesses a **weight** w_{uv} and
each neuron $u \in U$ possesses three (real-valued) state variables:

- the **network input** net_u,
- the **activation** act_u, and
- the **output** out_u.

Each input neuron $u \in U_{\mathrm{in}}$ also possesses a fourth (real-valued) state variable,

- the **external input** ex_u.

Furthermore, each neuron $u \in U$ possesses three functions:

- the **network input function** $f_{\mathrm{net}}^{(u)} : \mathbb{R}^{2|\,\mathrm{pred}(u)|+\kappa_1(u)} \to \mathbb{R}$,
- the **activation function** $f_{\mathrm{act}}^{(u)} : \mathbb{R}^{\kappa_2(u)} \to \mathbb{R}$, and
- the **output function** $f_{\mathrm{out}}^{(u)} : \mathbb{R} \to \mathbb{R}$,

which are used to compute the values of the state variables.

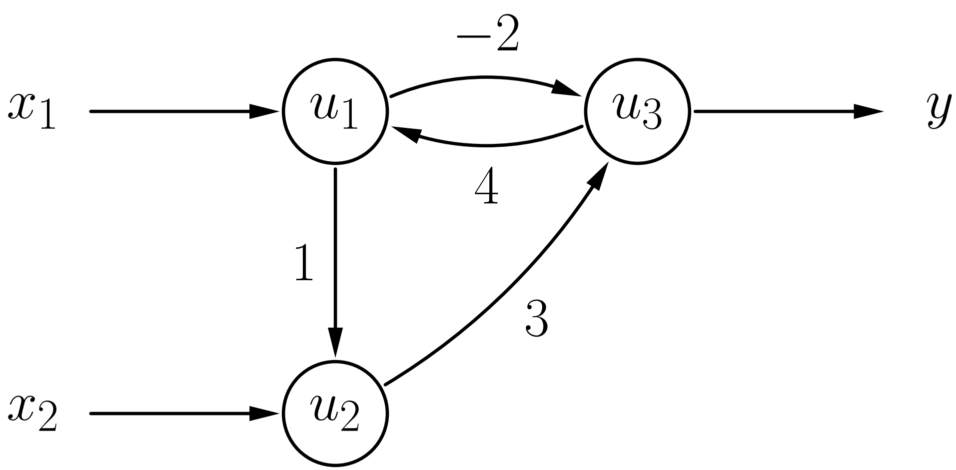
Types of (artificial) neural networks

- If the graph of a neural network is **acyclic**,
  it is called a **feed-forward network**.
- If the graph of a neural network contains **cycles** (backward connections),
  it is called a **recurrent network**.

Representation of the connection weights by a matrix

$$\begin{array}{cc}
\begin{array}{cccc} u_1 & u_2 & \dots & u_r \end{array} & \\
\begin{pmatrix}
w_{u_1u_1} & w_{u_1u_2} & \dots & w_{u_1u_r} \\
w_{u_2u_1} & w_{u_2u_2} & & w_{u_2u_r} \\
\vdots & & & \vdots \\
w_{u_ru_1} & w_{u_ru_2} & \dots & w_{u_ru_r}
\end{pmatrix} & \begin{array}{c} u_1 \\ u_2 \\ \vdots \\ u_r \end{array}
\end{array}$$

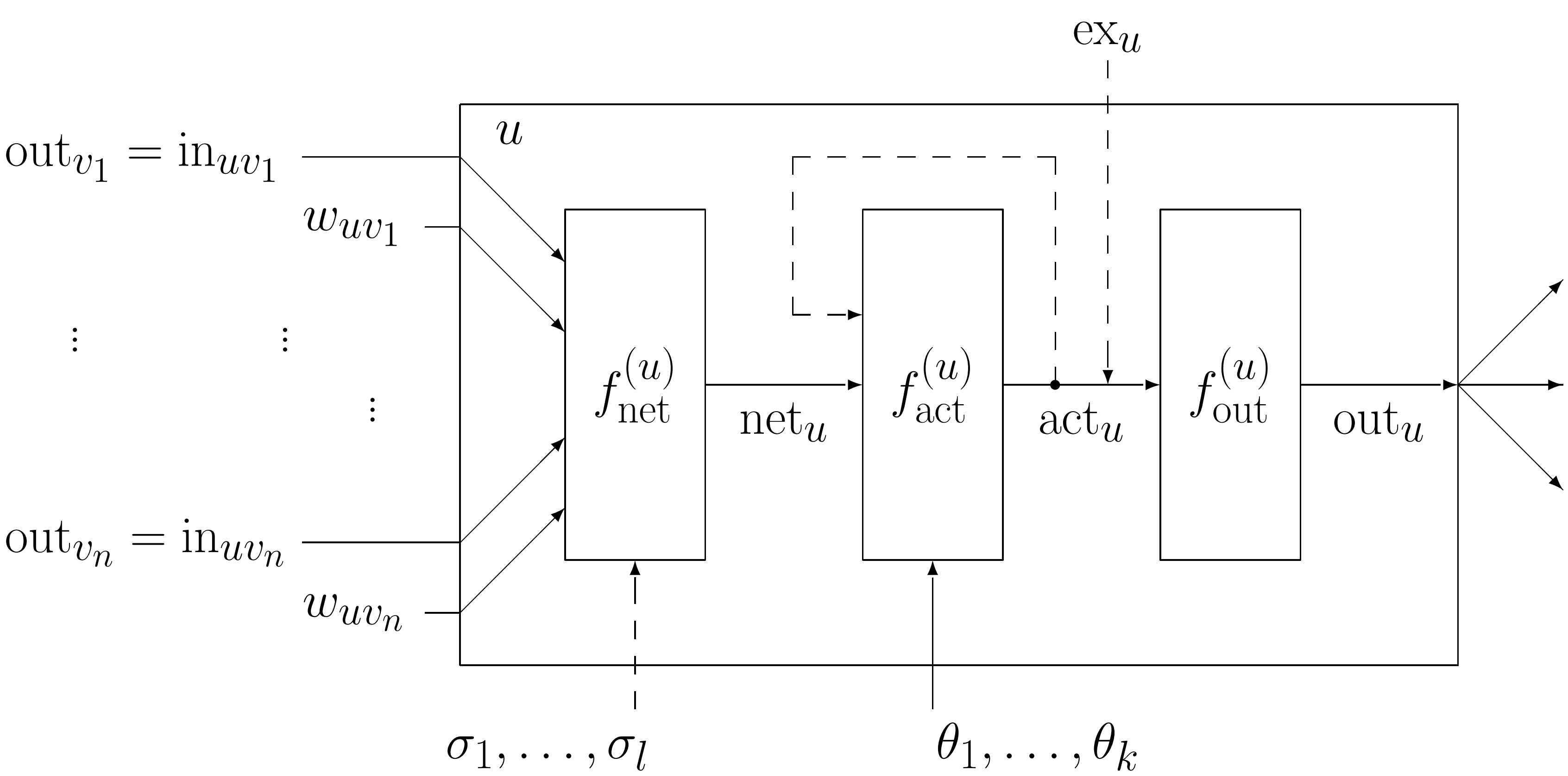
A simple recurrent neural network



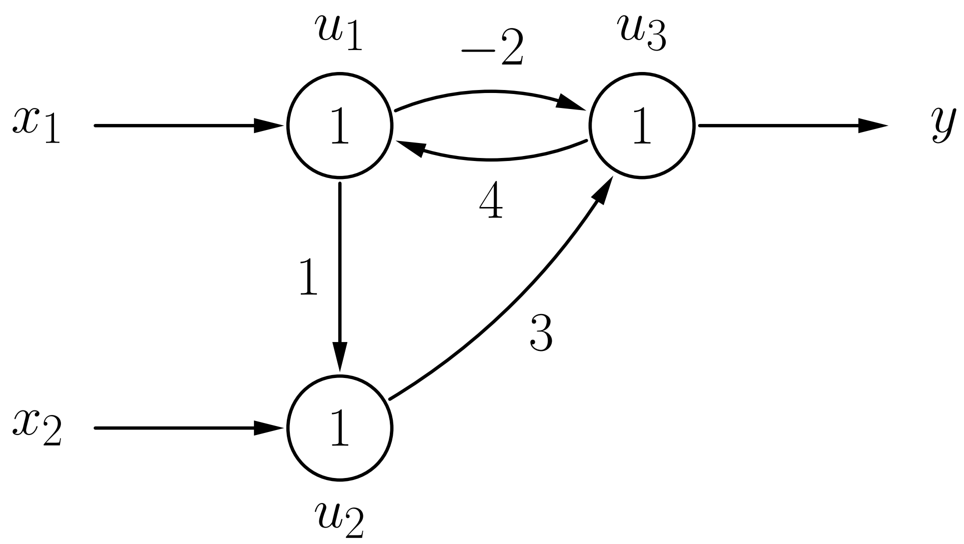
Weight matrix of this network

$$\begin{array}{cc} \begin{matrix} \quad u_1 & u_2 & u_3 \end{matrix} & \\ \begin{pmatrix} 0 & 0 & 4 \\ 1 & 0 & 0 \\ -2 & 3 & 0 \end{pmatrix} & \begin{matrix} u_1 \\ u_2 \\ u_3 \end{matrix} \end{array}$$

A generalized neuron is a simple numeric processor



General Neural Networks: Example



$$f_{\text{net}}^{(u)}(\vec{w}_u, \vec{\text{in}}_u) = \sum_{v \in \text{pred}(u)} w_{uv} \text{in}_{uv} = \sum_{v \in \text{pred}(u)} w_{uv} \, \text{out}_v$$

$$f_{\text{act}}^{(u)}(\text{net}_u, \theta) = \begin{cases} 1, & \text{if} \quad \text{net}_u \geq \theta, \\ 0, & \text{otherwise.} \end{cases}$$

$$f_{\text{out}}^{(u)}(\text{act}_u) = \text{act}_u$$

General Neural Networks: Example

Updating the activations of the neurons

	u_1	u_2	u_3	
input phase	**1**	**0**	**0**	
work phase	1	0	**0**	$\text{net}_{u_3} = -2$
	0	0	0	$\text{net}_{u_1} = 0$
	0	**0**	0	$\text{net}_{u_2} = 0$
	0	0	**0**	$\text{net}_{u_3} = 0$
	0	0	0	$\text{net}_{u_1} = 0$

- Order in which the neurons are updated: $u_3, u_1, u_2, u_3, u_1, u_2, u_3, \ldots$
- Input phase: activations and outputs of the initial state (first row)
- The activation of the currently neuron (bold) is calculated by considering the other neurons and weights.
- A stable state with a unique output is reached.

Updating the activations of the neurons

	u_1	u_2	u_3	
input phase	**1**	**0**	**0**	
work phase	1	0	**0**	$\text{net}_{u_3} = -2$
	1	**1**	0	$\text{net}_{u_2} = 1$
	0	1	0	$\text{net}_{u_1} = 0$
	0	1	**1**	$\text{net}_{u_3} = 3$
	0	**0**	1	$\text{net}_{u_2} = 0$
	1	0	1	$\text{net}_{u_1} = 4$
	1	0	**0**	$\text{net}_{u_3} = -2$

- Order in which the neurons are updated:
  $u_3, u_2, u_1, u_3, u_2, u_1, u_3, \ldots$
- No stable state is reached (oscillation of output).

Definition of learning tasks for a neural network

A **fixed learning task** L_{fixed} for a neural network with

- n input neurons, i.e. $U_{\text{in}} = \{u_1, \ldots, u_n\}$, and
- m output neurons, i.e. $U_{\text{out}} = \{v_1, \ldots, v_m\}$,

is a set of **training patterns** $l = (\vec{\imath}^{\,(l)}, \vec{o}^{\,(l)})$, each consisting of

- an **input vector** $\vec{\imath}^{\,(l)} = (\,\text{ex}_{u_1}^{(l)}, \ldots, \text{ex}_{u_n}^{(l)}\,)$ and
- an **output vector** $\vec{o}^{\,(l)} = (o_{v_1}^{(l)}, \ldots, o_{v_m}^{(l)})$.

A fixed learning task is solved, if for all training patterns $l \in L_{\text{fixed}}$ the neural network computes from the external inputs contained in the input vector $\vec{\imath}^{\,(l)}$ of a training pattern l the outputs contained in the corresponding output vector $\vec{o}^{\,(l)}$.

Solving a fixed learning task: Error definition

- Measure how well a neural network solves a given fixed learning task.
- Compute differences between desired and actual outputs.
- Do not sum differences directly in order to avoid errors canceling each other.
- Square has favorable properties for deriving the adaptation rules.

$$e = \sum_{l \in L_{\text{fixed}}} e^{(l)} = \sum_{v \in U_{\text{out}}} e_v = \sum_{l \in L_{\text{fixed}}} \sum_{v \in U_{\text{out}}} e_v^{(l)},$$

$$\text{where} \qquad e_v^{(l)} = \left(o_v^{(l)} - \text{out}_v^{(l)} \right)^2$$

Definition of learning tasks for a neural network

A **free learning task** L_{free} for a neural network with

- n input neurons, i.e. $U_{\text{in}} = \{u_1, \ldots, u_n\}$,

is a set of **training patterns** $l = (\vec{\imath}^{\,(l)})$, each consisting of

- an **input vector** $\vec{\imath}^{\,(l)} = (\,\text{ex}_{u_1}^{(l)}, \ldots, \text{ex}_{u_n}^{(l)}\,)$.

Properties:

- There is no desired output for the training patterns.
- Outputs can be chosen freely by the training method.
- Solution idea: **Similar inputs should lead to similar outputs.** (clustering of input vectors)

Normalization of the input vectors

- Compute expected value and standard deviation for each input:

$$\mu_k = \frac{1}{|L|} \sum_{l \in L} \mathrm{ex}_{u_k}^{(l)} \qquad \text{and} \qquad \sigma_k = \sqrt{\frac{1}{|L|} \sum_{l \in L} \left(\mathrm{ex}_{u_k}^{(l)} - \mu_k \right)^2},$$

- Normalize the input vectors to expected value 0 and standard deviation 1:

$$\mathrm{ex}_{u_k}^{(l)(\mathrm{neu})} = \frac{\mathrm{ex}_{u_k}^{(l)(\mathrm{alt})} - \mu_k}{\sigma_k}$$

- Avoids unit and scaling problems.